UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/986,248	11/08/2001	William Russell Belknap	SVL920010059US	5036

23373 7590 01/17/2006

SUGHRUE MION, PLLC
2100 PENNSYLVANIA AVENUE, N.W.
SUITE 800
WASHINGTON, DC 20037

EXAMINER	
BONSHOCK, DENNIS G	

ART UNIT	PAPER NUMBER
2173	

DATE MAILED: 01/17/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/986,248
Filing Date: November 08, 2001
Appellant(s): BELKNAP ET AL.

Brandon M. White (Registration No. 52,354)
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 10-04-2005 appealing from the Office action mailed 7-25-2005.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

No evidence is relied upon by the examiner in the rejection of the claims under appeal.

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

2. Claims 1-3, 13-15, 25, and 26 are rejected under 35 U.S.C. 102(e) as being anticipated by Halpern et al., patent #6,282,711, hereinafter Halpern.

3. With regard to claim 1, which teaches a method of requesting and processing a plurality of objects from a server, comprising: requesting a plurality of objects from the server, Halpern teaches, in column 3, lines 16-38 and in column 5, lines 5-51, a user selecting a plurality of objects from a server. With regard to claim 1, further teaching receiving a response message from the server, the response message containing the plurality of objects packed into the response message, Halpern teaches, in column 3, line 61 though column 4, line 5 and column 6, lines 1-28, the client receiving a package from the server containing the plurality of selected objects. With regard to claim 1, further teaching automatically unpacking the plurality of objects contained in the response message, Halpern teaches, in column 6, lines 44-64, and in column 4, lines 14-19, an automatic unpacking of objects that doesn't require user interaction.

4. With regard to claims 2, 14 and 26, which teach decompressing the plurality of unpacked objects, Halpern teaches, in column 6, lines 44-64, the automatic decompression of the transferred objects.

5. With regard to claims 3 and 15, which teach decompressing the plurality of unpacked objects automatically in response to receiving the response message, Halpern teaches, in column 6, lines 44-64, the automatic decompression of the transferred objects.

9. With regard to claim 13, which teaches a client processor, comprising: a communications module configured for receiving a response message from the server, the response message containing the plurality of objects packed into the response message, Halpern teaches, in column 3, line 61 though column 4, line 5 and column 6, lines 1-28, the client receiving a package from the server containing the plurality of selected objects. With regard to claim 13, further teaching automatically unpacking the plurality of objects contained in the response message, Halpern teaches, in column 6, lines 44-64, and in column 4, lines 14-19, an automatic unpacking of objects that doesn't require user interaction. With regard to claim 13, further teaching a browser coupled to the unpacking module, configured to present the plurality of unpacked objects to a user, Halpern further teaches, in column 4, line 54 through column 5, line 5, providing a display of the transfer system through the use of a browser.

11. With regard to claim 25, which teaches a computer readable medium for requesting and processing a plurality of objects from a server, comprising: program instructions for requesting a plurality of objects from the server, Halpern teaches, in

column 3, lines 16-38 and in column 5, lines 5-51, a user selecting a plurality of objects from a server. With regard to claim 25, further teaching program instructions for receiving a response message from the server, the response message containing the plurality of objects packed into the response message, Halpern teaches, in column 3, line 61 though column 4, line 5 and column 6, lines 1-28, the client receiving a package from the server containing the plurality of selected objects. With regard to claim 25, further teaching program instructions for automatically unpacking the plurality of objects contained in the response message, Halpern teaches, in column 6, lines 44-64, and in column 4, lines 14-19, an automatic unpacking of objects that doesn't require user interaction.

Claim Rejections - 35 USC § 103

14. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

15. Claims 4, 5, 16 and 17 rejected under 35 U.S.C. 103(a) as being unpatentable over Halpern.

16. With regard to claim 4 and 16, which teach requesting a plurality of objects comprises packing a plurality of requests for the plurality of objects into a packed request message and transmitting the packed request message to the server, Halpern teaches requests sent to the server for a plurality of objects (see column 3, lines 16-38 and column 5, lines 5-51), he however doesn't specifically specify if the requests are

sent individually for each object or as packed request. It would have been obvious to one of ordinary skill in the art, having the teachings of Halpern to provide the user with the option of sending the request to the server as either a single package or as a plurality of packages, similar to how Halpern offers the transfer of data between the server and the client (see column 3, line 61 through column 4, line 9 and in column 6, lines 17-28). One would have been motivated to make such a combination because in the bi-directional transfer system of Halpern, it would be beneficial, in terms of time saved in the case of lost objects, to provide the same optional packeting of objects in the client to server transfer.

17. With regard to claims 5 and 17, which teach requesting a plurality of objects comprises transmitting to the server separate requests for each of the plurality of objects, Halpern teaches requests sent to the server for a plurality of objects (see column 3, lines 16-38 and column 5, lines 5-51), he however doesn't specifically specify if the requests are sent individually for each object or as packed request. It would have been obvious to one of ordinary skill in the art, having the teachings of Halpern to provide the user with the option of sending the request to the server as either a single package or as a plurality of packages, similar to how Halpern offers the transfer of data between the server and the client (see column 3, line 61 through column 4, line 9 and in column 6, lines 17-28). One would have been motivated to make such a combination because in the bi-directional transfer system of Halpern, it would be beneficial, in terms of time saved in the case of lost objects, to provide the same optional packeting of objects in the client to server transfer.

18. Claims 6-10, 18-23, 27-29, 31, and 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Halpern and Feinman, patent #6,075,943.

19. With regard to claims 6, 18, and 27, Halpern teaches a system for the transfer of multiple objects between a server and a client and outputting a plurality of unpacked objects (see column 6, lines 1-67), but doesn't specifically teach outputting the plurality of objects in an order indicated in the response message. Feinman teaches a system for packaging up one or more applications for transfer between a server and a client (see column 2, lines 34-45) similar to that of Halpern, but further teaches, in column 3, line 43 through column 4, line 12, the outputting of applications having a certain order, as indicated by the server. It would have been obvious to one of ordinary skill in the art, having the teachings of Halpern and Feinman to include an ordering of objects, as did Feinman in the object transfer system of Halpern. One would have been motivated to make such a combination because this would provide an efficient means for allowing the server to dictate the order in which objects must be presented.

20. With regard to claims 7 and 19, which further teaches the plurality of unpacked objects being presented by a browser in the order the objects are output, Halpern further teaches, in column 4, line 54 through column 5, line 5, providing a display of the transfer system through the use of a browser.

6. With regard to claim 8, which teaches a method of transferring a plurality of objects from a server to a client comprising: receiving a request from the client for the plurality of objects, Halpern teaches, in column 6, lines 1-5, the request for a plurality of objects. With regard to claim 8, further teaching retrieving the plurality of requested

objects from one or more object stores, Halpern teaches, in column 5, lines 49-55 and in column 6, lines 1-5, the server retrieving the requested objects from a component pool. With regard to claim 8, further teaching automatically packing the retrieved plurality of objects into a response message, Halpern teaches, in column 5, lines 49-55 and in column 6, lines 1-15, the server retrieving the requested objects from a component pool and forms a customized non-binging set of files. With regard to claim 8, further teaching transmitting the response message to the client, Halpern teaches, in column 6, lines 17-19, the executable prepared by the packager being transmitted over a network to the client. Halpern teaches a system for the transfer of multiple objects between a server and a client and outputting a plurality of unpacked objects (see column 6, lines 1-67), but doesn't specifically teach the response message including an indicator of the order in which the packed objects are to be presented. Feinman teaches a system for packaging up one or more applications for transfer between a server and a client (see column 2, lines 34-45) similar to that of Halpern, but further teaches, in column 3, line 43 through column 4, line 12, the outputting of applications providing an indication of a certain order, as indicated by the server. The compression and decompression of the files done by compression and decompression programs (column 3, lines 7-43), where the automatic installations system builds a command for the remote submission, the command (indication) containing the name of the appropriate decompression program to run (which specifies the order to present data) (see column 5, lines 49-55). It would have been obvious to one of ordinary skill in the art, having the teachings of Halpern and Feinman to include an ordering of objects, as

did Feinman in the object transfer system of Halpern. One would have been motivated to make such a combination because this would provide an efficient means for allowing the server to dictate the order in which objects must be presented.

7. With regard to claims 9, 21, and 29, which teach automatically compressing the retrieved plurality of requested objects prior to packing the objects into the response message, Halpern teaches, in column 3, line 61 through column 4, line 5, the step of compressing and packaging the files together before transfer.

8. With regard to claims 10 and 22, which teaches automatically compressing the response message prior to transmitting the response message to the client, Halpern teaches, in column 3, line 61 through column 4, line 5, the step of compressing and packaging the files together before transfer.

10. With regard to claim 20, which teaches a server processor comprising: a module configured to receiving a request from the client for the plurality of objects, Halpern teaches, in column 6, lines 1-5, the request for a plurality of objects. With regard to claim 20, further teaching a processor configured for retrieving the plurality of requested objects from one or more object stores, Halpern teaches, in column 5, lines 49-55 and in column 6, lines 1-5, the server retrieving the requested objects from a component pool. With regard to claim 20, further teaching a module configured to automatically packing the retrieved plurality of objects into a response message, Halpern teaches, in column 5, lines 49-55 and in column 6, lines 1-15, the server retrieving the requested objects from a component pool and forms a customized non-binging set of files. With regard to claim 20, further teaching a module configured to transmit the response

message to the client, Halpern teaches, in column 6, lies 17-19, the executable prepared by the packager being transmitted over a network to the client. Halpern teaches a system for the transfer of multiple objects between a server and a client and outputting a plurality of unpacked objects (see column 6, lines 1-67), but doesn't specifically teach the response message including an indicator of the order in which the packed objects are to be presented. Feinman teaches a system for packaging up one or more applications for transfer between a server and a client (see column 2, lines 34-45) similar to that of Halpern, but further teaches, in column 3, line 43 through column 4, line 12, the outputting of applications providing an indication of a certain order, as indicated by the server. The compression and decompression of the files done by compression and decompression programs (column 3, lines 7-43), where the automatic installations system builds a command for the remote submission, the command (indication) containing the name of the appropriate decompression program to run (which specifies the order to present data) (see column 5, lines 49-55). It would have been obvious to one of ordinary skill in the art, having the teachings of Halpern and Feinman to include an ordering of objects, as did Feinman in the object transfer system of Halpern. One would have been motivated to make such a combination because this would provide an efficient means for allowing the server to dictate the order in which objects must be presented.

21. With regard to claims 23 and 32, Halpern teaches a system for the transfer of multiple objects between a server and a client and outputting a plurality of unpacked objects (see column 6, lines 1-67), but doesn't specifically teach the retrieved objects

being packed into the response message in a designated order. Feinman teaches a system for packaging up one or more applications for transfer between a server and a client (see column 2, lines 34-45) similar to that of Halpern, but further teaches, in column 3, line 43 through column 4, line 12, the outputting of applications having a certain order, as indicated by the server. It would have been obvious to one of ordinary skill in the art, having the teachings of Halpern and Feinman to include an ordering of objects, as did Feinman in the object transfer system of Halpern. One would have been motivated to make such a combination because this would provide an efficient means for allowing the server to dictate the order in which objects must be presented.

12. With regard to claim 28, which teaches a method of transferring a plurality of objects from a server to a client comprising: program instructions for receiving a request from the client for the plurality of objects, Halpern teaches, in column 6, lines 1-5, the request for a plurality of objects. With regard to claim 28, further teaching program instructions for retrieving the plurality of requested objects from one or more object stores, Halpern teaches, in column 5, lines 49-55 and in column 6, lines 1-5, the server retrieving the requested objects from a component pool. With regard to claim 28, further teaching program instructions for automatically packing the retrieved plurality of objects into a response message, Halpern teaches, in column 5, lines 49-55 and in column 6, lines 1-15, the server retrieving the requested objects from a component pool and forms a customized non-binging set of files. With regard to claim 28, further teaching program instructions for transmitting the response message to the client, Halpern teaches, in column 6, lines 17-19, the executable prepared by the packager

being transmitted over a network to the client. Halpern teaches a system for the transfer of multiple objects between a server and a client and outputting a plurality of unpacked objects (see column 6, lines 1-67), but doesn't specifically teach the response message including an indicator of the order in which the packed objects are to be presented. Feinman teaches a system for packaging up one or more applications for transfer between a server and a client (see column 2, lines 34-45) similar to that of Halpern, but further teaches, in column 3, line 43 through column 4, line 12, the outputting of applications providing an indication of a certain order, as indicated by the server. The compression and decompression of the files done by compression and decompression programs (column 3, lines 7-43), where the automatic installations system builds a command for the remote submission, the command (indication) containing the name of the appropriate decompression program to run (which specifies the order to present data) (see column 5, lines 49-55). It would have been obvious to one of ordinary skill in the art, having the teachings of Halpern and Feinman to include an ordering of objects, as did Feinman in the object transfer system of Halpern. One would have been motivated to make such a combination because this would provide an efficient means for allowing the server to dictate the order in which objects must be presented.

13. With regard to claim 31, which teaches a method of transferring a plurality of objects from a server to a client comprising: receiving a request from the client for the plurality of objects, Halpern teaches, in column 6, lines 1-5, the request for a plurality of objects. With regard to claim 31, further teaching retrieving the plurality of requested

objects from an object stores, Halpern teaches, in column 5, lines 49-55 and in column 6, lines 1-5, the server retrieving the requested objects from a component pool. With regard to claim 31, further teaching packing the retrieved plurality of objects into a response message, Halpern teaches, in column 5, lines 49-55 and in column 6, lines 1-15, the server retrieving the requested objects from a component pool and forms a customized non-binging set of files. With regard to claim 31, further teaching transmitting the response message to the client, Halpern teaches, in column 6, lines 17-19, the executable prepared by the packager being transmitted over a network to the client. Halpern teaches a system for the transfer of multiple objects between a server and a client and outputting a plurality of unpacked objects (see column 6, lines 1-67), but doesn't specifically teach the response message including an indicator of the order in which the packed objects are to be presented. Feinman teaches a system for packaging up one or more applications for transfer between a server and a client (see column 2, lines 34-45) similar to that of Halpern, but further teaches, in column 3, line 43 through column 4, line 12, the outputting of applications providing an indication of a certain order, as indicated by the server. The compression and decompression of the files done by compression and decompression programs (column 3, lines 7-43), where the automatic installations system builds a command for the remote submission, the command (indication) containing the name of the appropriate decompression program to run (which specifies the order to present data) (see column 5, lines 49-55). It would have been obvious to one of ordinary skill in the art, having the teachings of Halpern and Feinman to include an ordering of objects, as did Feinman in the object transfer

system of Halpern. One would have been motivated to make such a combination because this would provide an efficient means for allowing the server to dictate the order in which objects must be presented.

(10) Response to Argument

Claims 1-3, 13-15, and 25-26:

With respect to the arguments directed at the group of claims including Claims **1-3, 13-15, and 25-26** the Appellant's arguments are focused on the limitations regarding automatically unpacking the plurality of objects. More specifically, as stated from representative Claim 1, the limitation argued is:

"...automatically unpacking the plurality of objects contained in the response message."

Since the interpretation of the limitation is the basis for the arguments, the Examiner's interpretation is now given. The claim, as interpreted by the examiner, pertains to an unpacking (expanding from a compacted form) a number of objects in a response message, where this expanding from a compacted state is done automatically (without a user manually unpacking each object). As stated in the eighth paragraph of MPEP 2101[R2].II.C.,

"*Office personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure.* In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023,1027-28 (Fed. Cir. 1997)."

Based on the interpretation of the claim limitations being argued, the Examiner will now explain how the teachings of the Halpern reference are within the scope of these limitations.

Halpern discloses in column 3, lines 16-38 and in column 5, lines 5-51, a user selecting a plurality of objects from a server; in column 3, line 61 though column 4, line 5 and column 6, lines 1-28, the client receiving a package from the server containing the plurality of selected objects; and in column 6, lines 44-64, and in column 4, lines 14-19, an automatic unpacking of objects that doesn't require user interaction. Though the user may initiate the installation process by clicking on and executable (setup.exe or install.exe), the actual unpacking of objects is done automatically. This can further be shown in column 3, line 61 through column 4, line 5, that teaches the package being a compressed (packed), self-extracting (an program file that un-compresses it's own compressed files) executable.

Several definitions are presented below from the Microsoft Computer Dictionary, Fifth Edition, to further clarify both the reference and the claimed invention.

pack *vb.* To store information in a more compact form. Packing eliminates unnecessary spaces and other such characters and may use other special

methods of compressing data as well. It is used by some programs to minimize storage requirements.

compress *vb.* To reduce the size of a set of data, such as a file or communications message, so that it can be stored in less space or transmitted with less bandwidth. Data can be compressed by removing repeated patterns of bits and replacing them with some form of summary that takes up less space restoring the repeated patterns decompresses the data. Lossless compression methods must be used for text, code, and numeric data files; lossy compression may be used for video and sound files.

self-extracting file *n.* An executable program file that contains one or more compressed text of data files. When a user runs the program, it uncompresses the compressed files and stores them on the user's hard drive.

The examiner will now address the individual arguments and statements made by Appellant.

From page 14 of the Appeal Brief, from the third paragraph, the Appellant argues that "Claim 1 requires "automatically unpacking the plurality of objects contained in the response message." It is respectfully submitted that Halpern does not disclose this limitation and therefore does not anticipate claim 1."

The examiner respectfully contends that Halpern teaches, in column 5, lines 43-55 and in column 6, lines 1-27, a user selecting desired options to be delivered, and in response to the request receiving a compressed filed with is received at the client end.

The file is received as a self-extracting executable, i.e. a file that contains one or more compressed text or data files that when run, un-compresses (unpacks) the compressed (packed) files (see Microsoft Computer Dictionary definition above) (see column 6, line 47). This self-extracting executable includes program and data files and a client installer program as well as decompression and auto-start utilities, the user "may" simply execute the received setup.exe or install.exe file to immediately install the application and options which were selected. It is further stated that the above "client installer program" may be configured to permit the contents of the delivered package to be installed without further user intervention (see column 6, lines 44-56). Halpern also teaches an embodiment in which application programs are installed from the component pool without any interaction between the user and the options manager (see column 6, lines 29-35).

In summary Halpern teaches a system in which the response message, comprising compressed (packed) data, is received at the client end where it is either automatically installed, or waits for a user to execute the received setup.exe or install.exe file to install. But regardless of how the install is initiated, there exists a program the automatically decompresses (unpacks) the compressed (packed) objects.

From page 15 of the Appeal Brief, from the second paragraph, the Appellant argues that "In stating that the client installer program "may be configured to permit installation without "further user interaction", Halpern makes clear that the user must at

least take the action of executing the received setup.exe or install.exe files. Because Halpern discloses that a user must interact with the system to setup or install files, Halpern does not disclose "automatically unpacking" as recited in claim 1."

The examiner respectfully contends that Halpern teaches an embodiment in which application programs are installed (where objects must be unpacked to be installed) from the component pool without any interaction between the user and the options manager (see column 6, lines 29-35). The claim language used by the client is "automatically unpacking" not "automatically initiating unpacking" or "automatically initiating installation and unpacking". This "automatically unpacking" is a process of decompressing objects automatically, without user interaction. The user does not manually pull apart objects separating them from the compressed file. The "self-extracting executable" of Halpern is a file that contains one or more compressed text or data files that when run, un-compresses (unpacks) the compressed (packed) files (see Microsoft Computer Dictionary definition above) (see column 6, line 47). This "self-extracting executable" relieves the user from manually pull objects from the compressed file.

Claims 4-5 and 16-17:

From pages 16 and 17 of the Appeal Brief, from the fourth paragraph on page 16, the Appellant argues that "the Examiner asserts without support that "[o]ne would have been motivated [to combine Halpern and Feinman because] it would be beneficial , in terms of time save in the case lost objects, to provide the same optional packeting of

objects in the client to server transfer." However, the motivation to modify the references must be taught or suggested by the prior art... The Examiner fails to cite any support in the prior art to modify Halpern."

The examiner respectfully contends that first of all Feinman is not relied upon for this Rejection of claims 4-5 and 16-17 under 103 (a). In support for this assertion the examiner has previously stated that Halpern offers the transfer of data between the server and the client (see column 3, line 61 through column 4, line 9 and in column 6, lines 17-28). This transfer is via a packet that contains a number of packets transmittable via a packetization transport protocol. Where it is further stated (in column 4, lines 5-10) that individual packet can be transmitted. If this packetization transport protocol is used in one direction it obviously would be used in the other direction.

Claims 6-7, 18-19, and 27:

With respect to the arguments directed at the group of claims including Claims **6-7, 18-19, and 27** the Appellant's arguments are focused on the limitations regarding outputting objects in a specified order. More specifically, as stated from representative Claim 1, the limitation argued is:

"...outputting the plurality of unpacked objects in an order indicated in the response message."

Since the interpretation of the limitation is the basis for the arguments, the Examiner's interpretation is now given. The claim, as interpreted by the examiner, pertains to outputting objects in an order that is provided by the response message, this

order provided in any means. As stated in the eighth paragraph of MPEP 2101[R2].II.C.,

> "*Office personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure.* In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023,1027-28 (Fed. Cir. 1997)."

From page 19 of the Appeal Brief, from the first paragraph, the Appellant argues that "The Examiner relies on the disclosure in Feinman at col. 3, line 43 to col. 4, line 12 to provide the limitation in claim 6 of "outputting the plurality of unpacked objects in an order indicated in the response message." However, this portion of Feinman does not disclose the limitation of claim 6."

The examiner respectfully contends that Feinman teaches a system for packaging up one or more applications for transfer between a server and a client (see column 2, lines 11-45) similar to that of Halpern, but further teaches, in column 3, line 43 through column 4, line 12, that once the application programs have been packed up, they are identified by the time the each application program is to be delivered, with this information being stored in a "sequential file". The "sequential file" building an order of application program delivery. Furthermore, there must exist some inherent order to any list.

From page 20 of the Appeal Brief, from the first paragraph, the Appellant argues that "Nothing in Feinman teaches or suggests including the sequential file in a response message or otherwise including any indicator in the response message for indicating any output order"

The examiner respectfully contends that Feinman teaches packing up a list of files to be transmitted to the client in a sequential file and further including a time that each application program is to be delivered in this file (see column 3, line 12 through column 4, line 12). The "sequential file" building an order of application program delivery. Furthermore, there must exist some inherent order to any list.

Claims 8-10, 20-23, 28-29, and 31-32:

With respect to the arguments directed at the group of claims including Claims **6-7, 18-19, and 27** the Appellant's arguments are focused on the limitations regarding including an order to be presented in the response message. More specifically, as stated from representative Claim 1, the limitation argued is:
"...the response message includes an indicator of the order in which the packed objects are to be presented".

Since the interpretation of the limitation is the basis for the arguments, the Examiner's interpretation is now given. The claim, as interpreted by the examiner, pertains to presenting objects in an order that is provided by the response message, this order provided in any means, and presenting encompassing any thing from

installing, displaying, delivering, etc. As stated in the eighth paragraph of MPEP 2101[R2].II.C.,

> "*Office personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure.* In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023,1027-28 (Fed. Cir. 1997)."

From page 21 of the Appeal Brief, from the first paragraph, the Appellant argues that "Feinman does not teach or suggest a method "wherein the response message includes an indicator of the order in which the packed objects are to be presented"

The examiner respectfully contends that Feinman teaches a system for packaging up one or more applications for transfer between a server and a client (see column 2, lines 11-45) similar to that of Halpern, but further teaches, in column 3, line 43 through column 4, line 12, that once the application programs have been packed up, they identified by the time the each application program is to be delivered, with this information being stored in a "sequential file". The "sequential file" building an order of application program delivery. Furthermore, there must exist some inherent order to any list.

The statement regarding the compression and decompression of the files done by compression and decompression programs (column 3, lines 7-43) was added to show how actual presentation is accomplished.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Conferees:

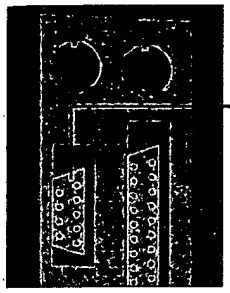
Dennis G. Bonshock
~~September 26, 2005~~

John W. Cabeca
Supervisory Patent Examiner
~~September 26, 2005~~ 01/06/06

Kristine Kincaid
Kristine Kincaid
Supervisory Patent Examiner
~~September 26, 2005~~

componentware *n. See* component software.

COM port or **comm port** *n.* Short for communications port, the logical address assigned by MS-DOS (versions 3.3 and later) and Microsoft Windows (including Windows 9x and Windows NT) to each of the four serial ports on an IBM Personal Computer or a PC compatible. COM ports also have come to be known as the actual serial ports on a PC's CPU where peripherals, such as printers, scanners, and external modems, are plugged in. See the illustration. *See also* COM (definition 1), input/output port, serial port.



COM port.

composite display *n.* A display, characteristic of television monitors and some computer monitors, that is capable of extracting an image from a composite signal (also called an *NTSC signal*). A composite display signal carries on one wire not only the coded information required to form an image on the screen but also the pulses needed to synchronize horizontal and vertical scanning as the electron beam sweeps back and forth across the screen. Composite displays can be either monochrome or color. A composite color signal combines the three primary video colors (red, green, and blue) in a color burst component that determines the shade of color displayed on the screen. Composite color monitors are less readable than either monochrome monitors or the RGB color monitors that use separate signals (and wires) for the red, green, and blue components of the image. *See also* color burst, color monitor, monochrome display, NTSC, RGB monitor.

composite key *n.* A key whose definition consists of two or more fields in a file, columns in a table, or attributes in a relation.

composite video display *n.* A display that receives all encoded video information (including color, horizontal synchronization, and vertical synchronization) in one signal. A composite video signal under NTSC (National Television System Committee) standards is generally required for television sets and videotape recorders. *See also* NTSC. *Compare* RGB monitor.

compound document *n.* A document that contains different types of information, each type created with a different application; for example, a report containing both charts (created with a spreadsheet) and text (created with a word processor) is a compound document. Although a compound document is visually a single, seamless unit, it is actually formed of discrete objects (blocks of information) that are created in their own applications. These objects can either be physically *embedded* in the destination document, or they can be *linked* to it while remaining in the originating file. Both embedded and linked objects can be edited. Linked objects, however, can be updated to reflect changes made to the source file. *See also* ActiveX, OLE, OpenDoc.

compound statement *n.* A single instruction composed of two or more individual instructions.

compress[1] *n.* A proprietary UNIX utility for reducing the size of data files. Files compressed with this utility have the extension .Z added to their names.

compress[2] *vb.* To reduce the size of a set of data, such as a file or a communications message, so that it can be stored in less space or transmitted with less bandwidth. Data can be compressed by removing repeated patterns of bits and replacing them with some form of summary that takes up less space; restoring the repeated patterns decompresses the data. Lossless compression methods must be used for text, code, and numeric data files; lossy compression may be used for video and sound files. *See also* lossless compression, lossy compression.

compressed digital video *n. See* CDV (definition 1).

compressed disk *n.* A hard disk or floppy disk whose apparent capacity to hold data has been increased through the use of a compression utility, such as Stacker or Double Space. *See also* data compression.

compressed drive *n.* A hard disk whose apparent capacity has been increased through the use of a compression utility, such as Stacker or Double Space. *See also* compressed disk, data compression.

compressed file *n.* A file whose contents have been compressed by a special utility program so that it occupies less space on a disk or other storage device than in its uncompressed (normal) state. *See also* installation program, LHARC, PKUNZIP, PKZIP, utility program.

Compressed Read-Only File System *n. See* cramfs.

P

p *prefix* *See* pico-.

P *prefix* *See* peta-.

P2P or P-to-P *n.* An Internet-based networking option in which two or more computers connect directly to each other to communicate and share files without use of a central server. Interest in P2P networking blossomed with the introduction of Napster and Gnutella. Short for Peer-to-Peer. *See also* peer-to-peer architecture, peer-to-peer communications.

P3P *n.* Acronym for Platform for Privacy Preferences. An open W3C protocol that allows Internet users to control the type of personal information that is collected by the Web sites they visit. P3P uses User Agents built into browsers and Web applications to allow P3P-enabled Web sites to communicate privacy practices to users before they log on to the Web site. P3P compares the Web site's privacy policies with the user's personal set of privacy preferences, and it reports any disagreements to the user.

P5 *n.* Intel's internal working name for the Pentium microprocessor. Although it was not intended to be used publicly, the name P5 leaked out to the computer-industry trade press and was commonly used to reference the microprocessor before it was released. *See also* 586, Pentium.

pack *vb.* To store information in a more compact form. Packing eliminates unnecessary spaces and other such characters and may use other special methods of compressing data as well. It is used by some programs to minimize storage requirements.

package *n.* **1.** A computer application consisting of one or more programs created to perform a particular type of work—for example, an accounting package or a spreadsheet package. **2.** In electronics, the housing in which an electronic component is packaged. *See also* DIP. **3.** A group of classes or interfaces and a keyword in the Java programming language. Packages are declared in Java by using the "package" keyword. *See also* class, declare, interface (definition 1), keyword.

packaged software *n.* A software program sold through a retail distributor, as opposed to custom software. *See also* canned software.

packed decimal *adj.* A method of encoding decimal numbers in binary form that maximizes storage space by using each byte to represent two decimal digits. When signed decimal numbers are stored in packed decimal format, the sign appears in the rightmost four bits of the rightmost (least significant) byte.

packet *n.* **1.** A unit of information transmitted as a whole from one device to another on a network. **2.** In packet-switching networks, a transmission unit of fixed maximum size that consists of binary digits representing both data and a header containing an identification number, source and destination addresses, and sometimes error-control data. *See also* packet switching.

packet assembler and disassembler *n.* *See* packet assembler/disassembler.

packet assembler/disassembler *n.* An interface between non-packet-switching equipment and a packet-switching network. *Acronym:* PAD.

packet filtering *n.* The process of controlling network access based on IP addresses. Firewalls will often incorporate filters that allow or deny users the ability to enter or leave a local area network. Packet filtering is also used to accept or reject packets such as e-mail, based on the origin of the packet, to ensure security on a private network. *See also* firewall, IP address, packet (definition 1).

packet flooding *n.* A technique employed in a number of DoS (denial of service) attacks in which a flood of packets of data are sent to a target server, overwhelming the computer and rendering it unable to respond to legitimate network requests. Examples of specific types of packet flooding include smurf attacks and SYN flood attacks. *See also* DoS, packet, smurf attack, SYN flood.

packet header *n.* The portion of a data packet that precedes the body (data). The header contains data, such as

seed *n.* A starting value used in generating a sequence of random or pseudorandom numbers. *See also* random number generation.

seek *n.* The process of moving the read/write head in a disk drive to the proper site, typically for a read or write operation.

seek time *n.* The time required to move a disk drive's read/write head to a specific location on a disk. *See also* access time (definition 2).

segment *n.* A section of a program that, when compiled, occupies a contiguous address space and that is usually position independent; that is, it can be loaded anywhere in memory. With Intel-based microcomputers, a native-mode segment is a logical reference to a 64-KB contiguous portion of RAM in which the individual bytes are accessed by means of an offset value. Collectively, the segment:offset values reference a single physical location in RAM. *See also* overlay[1] (definition 1), real mode, segmentation.

segmentation *n.* The act of breaking up a program into several sections, or segments. *See also* segment.

segmented addressing architecture *n.* A memory-access technique typified by Intel 80x86 processors. Memory is divided into 64-KB segments in this architecture for addressing locations under the 16-bit address scheme; 32-bit schemes can address memory in segments as large as 4 GB. *Also called:* segmented instruction addressing, segmented memory architecture. *Compare* linear addressing architecture.

segmented address space *n.* An address space that is logically divided into chunks called segments. To address a given location, a program must specify both a segment and an offset within that segment. (The offset is a value that references a specific point within the segment, based on the beginning of the segment.) Because segments may overlap, addresses are not unique; there are many logical ways to access a given physical location. The Intel 80x86 real-mode architecture is segmented; most other microprocessor architectures are flat. *See also* segment. *Compare* flat address space.

segmented instruction addressing *n. See* segmented addressing architecture.

segmented memory architecture *n. See* segmented addressing architecture.

select *vb.* **1.** In general computer use, to specify a block of data or text on screen by highlighting it or otherwise marking it with the intent of performing some operation on it. **2.** In database management, to choose records according to a specified set of criteria. *See also* sort. **3.** In information processing, to choose from a number of options or alternatives, such as subroutines or input/output channels.

selected cell *n. See* active cell.

selection *n.* **1.** In applications, the highlighted portion of an on-screen document. **2.** In communications, the initial contact made between a computer and a remote station receiving a message. **3.** In programming, a conditional branch. *See also* conditional branch.

selective calling *n.* The capability of a station on a communications line to designate the station that is to receive a transmission.

selector channel *n.* An input/output data transfer line used by one high-speed device at a time.

selector pen *n. See* light pen.

select query *n.* A query that asks a question about the data stored in your tables and returns a result set in the form of a datasheet, all without changing the data.

self-adapting *adj.* The ability of systems, devices, or processes to adjust their operational behavior to environmental conditions.

self-checking digit *n.* A digit, appended to a number during its encoding, whose function is to confirm the accuracy of the encoding. *See also* checksum, parity bit.

self-clocking *n.* A process in which timing signals are inserted into a data stream rather than being provided by an external source, such as in phase encoding.

self-documenting code *n.* Program source code that, through its use of a high-level language and descriptive identifiers, can be understood by other programmers without the need for additional comments.

self-extracting archive *n. See* self-extracting file.

self-extracting file *n.* An executable program file that contains one or more compressed text or data files. When a user runs the program, it uncompresses the compressed files and stores them on the user's hard drive. See the illustration.